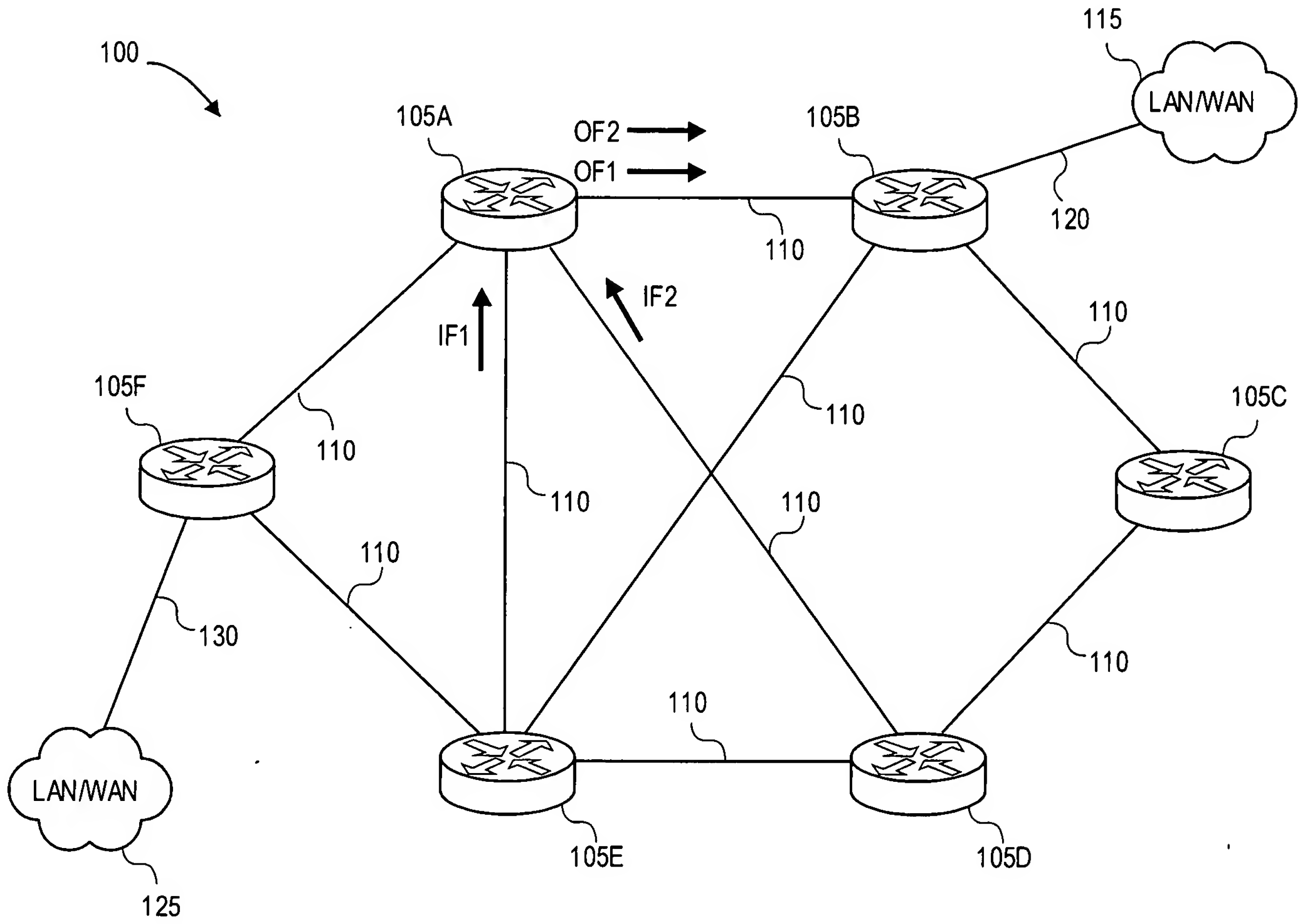

100
105A
OF2
OF1
105B
115
LAN/WAN
120
110
IF2
IF1
110
105F
110
110
105C
110
110
110
130
110
LAN/WAN
110
105E
105D
125

FIG. 1

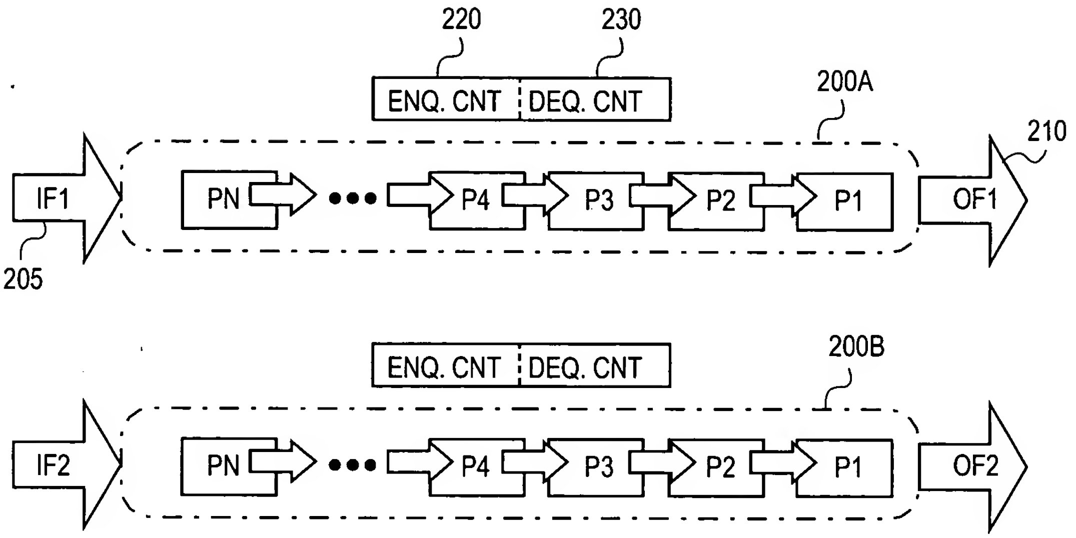
220
230
ENQ. CNT
DEQ. CNT
200A
210
IF1
PN
P4
P3
P2
P1
OF1
205
ENQ. CNT
DEQ. CNT
200B
IF2
PN
P4
P3
P2
P1
OF2

FIG. 2

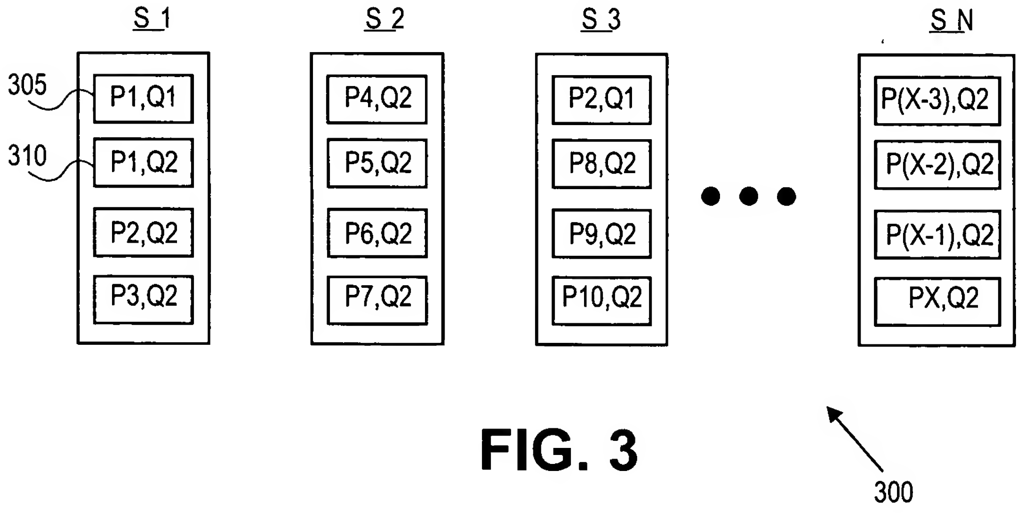
S 1
S 2
S 3
S N
305
310
P1,Q1
P1,Q2
P2,Q2
P3,Q2
P4,Q2
P5,Q2
P6,Q2
P7,Q2
P2,Q1
P8,Q2
P9,Q2
P10,Q2
P(X-3),Q2
P(X-2),Q2
P(X-1),Q2
PX,Q2
300

FIG. 3

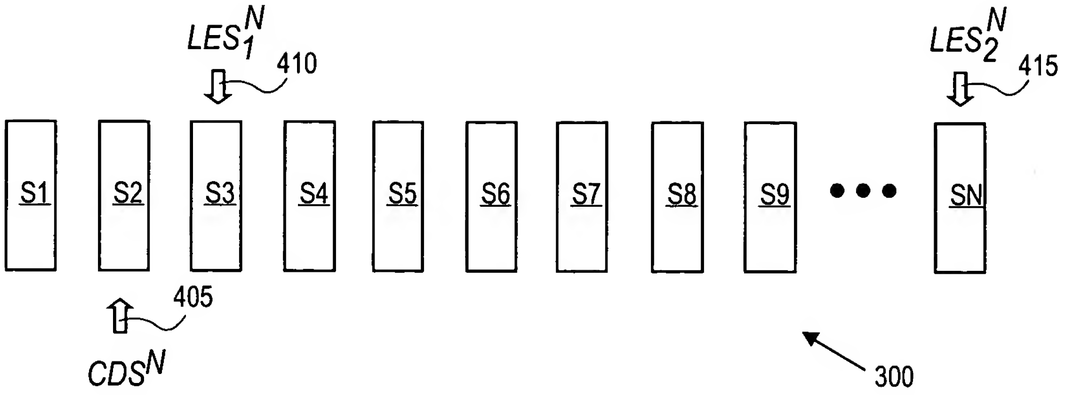
LES$_1^N$
410
LES$_2^N$
415
S1
S2
S3
S4
S5
S6
S7
S8
S9
SN
405
CDS^N
300

FIG. 4

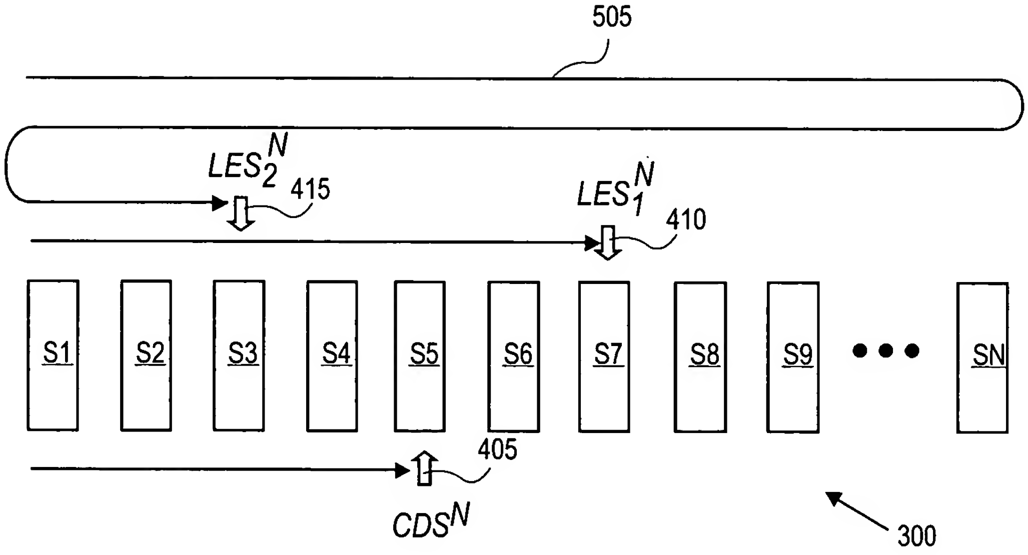
505
LES$_2^N$
415
LES$_1^N$
410
S1
S2
S3
S4
S5
S6
S7
S8
S9
SN
405
CDS^N
300

FIG. 5

600

610

```
enqueue( queue i)
{
        if (enqueue_count == dequeue_count)  // **check #1**
        {
                $LES_i^N = CDS^N$;                // queue is empty
        }
        else if ((($CDS^N$- $LES_i^N$) mod N) < M)  // **check #2**
        {
                $LES_i^N = CDS^N$;                // queue is empty and dequeue count is
                                                  //  lagging
        }

        $LES_i^N$ = ($LES_i^N$ + j) mod N;        // calculate where to enqueue the packet
                                                  // value j < M depends on queuing scheme
                                                  // Note: $LES_i^N$ may increase, $CDS^N$ unchanged
        if ((($CDR^N$- $LSR_i^N$) mod N) < M)     // **check #3**
        {
                Drop packet                       // queue has overflowed
                $LES_i^N$ = ($LES_i^N$ – j) mod N;   // reset $LES_i$ to old value
        }
        else
        {
                Enqueue the packet
        }
}
```

605

```
while (1)
{
        enqueue(i);                       // call enqueue routine for queue
                                          // value $LES_i$ may increase
        Perform some dequeues
        Perform enqueues on other queues
        $CDS^N$ = ($CDS^N$ + k) mod N;    // where k >= 0, k depends on how many rounds
                                          // have completed dequeues
                                          // Note: $LES_i^N$ unchanged, $CDS^N$ may increase
}
```

FIG. 6

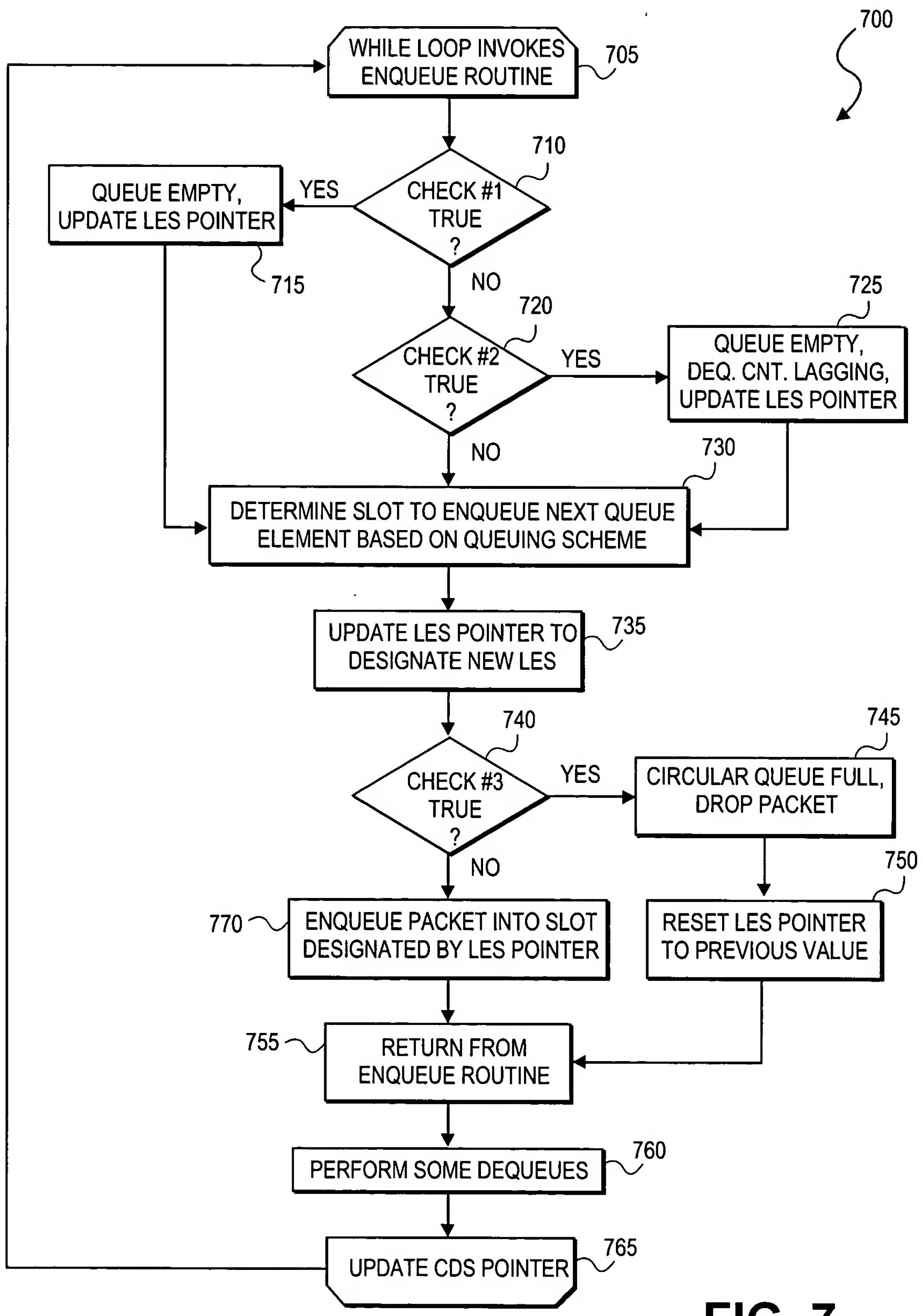
700
WHILE LOOP INVOKES ENQUEUE ROUTINE
705
CHECK #1 TRUE ?
710
YES
QUEUE EMPTY, UPDATE LES POINTER
715
NO
CHECK #2 TRUE ?
720
YES
QUEUE EMPTY, DEQ. CNT. LAGGING, UPDATE LES POINTER
725
NO
730
DETERMINE SLOT TO ENQUEUE NEXT QUEUE ELEMENT BASED ON QUEUING SCHEME
UPDATE LES POINTER TO DESIGNATE NEW LES
735
CHECK #3 TRUE ?
740
YES
CIRCULAR QUEUE FULL, DROP PACKET
745
NO
750
770
ENQUEUE PACKET INTO SLOT DESIGNATED BY LES POINTER
RESET LES POINTER TO PREVIOUS VALUE
755
RETURN FROM ENQUEUE ROUTINE
PERFORM SOME DEQUEUES
760
UPDATE CDS POINTER
765

FIG. 7

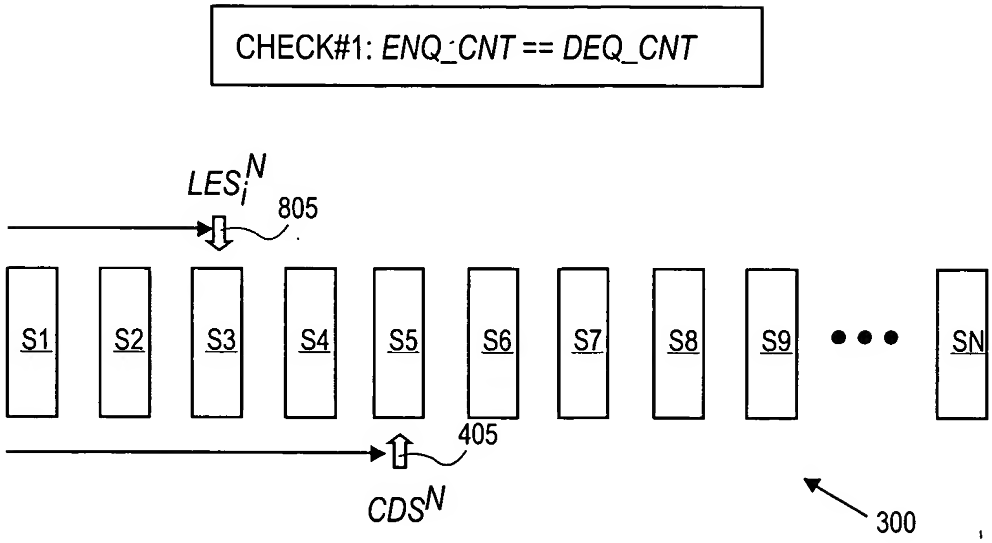
CHECK#1: ENQ_CNT == DEQ_CNT
LES_i^N
805
S1
S2
S3
S4
S5
S6
S7
S8
S9
SN
405
CDS^N
300

FIG. 8

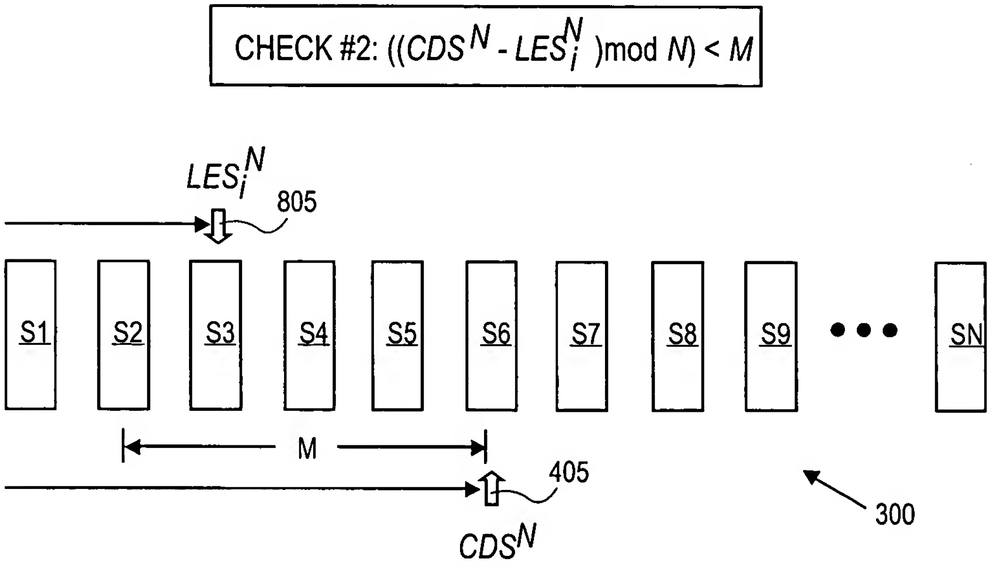
CHECK #2: ((CDS^N - LES_i^N)mod N) < M
LES_i^N
805
S1
S2
S3
S4
S5
S6
S7
S8
S9
SN
M
405
CDS^N
300

FIG. 9

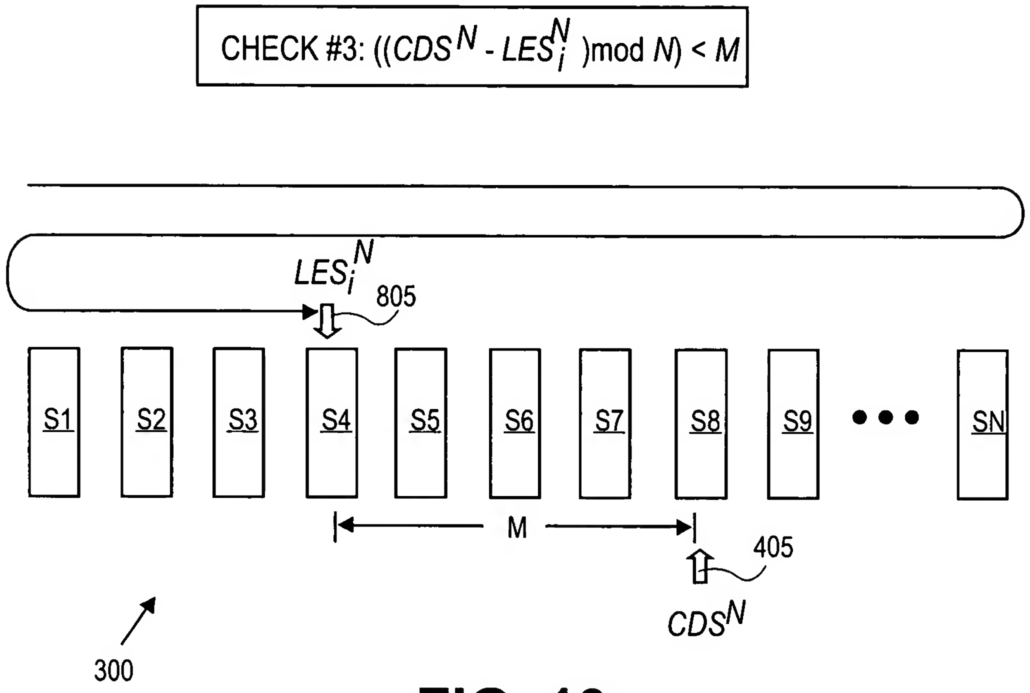
CHECK #3: $((CDS^N - LES_i^N)\text{mod } N) < M$
LES_i^N
805
S1
S2
S3
S4
S5
S6
S7
S8
S9
SN
M
405
CDS^N
300

FIG. 10

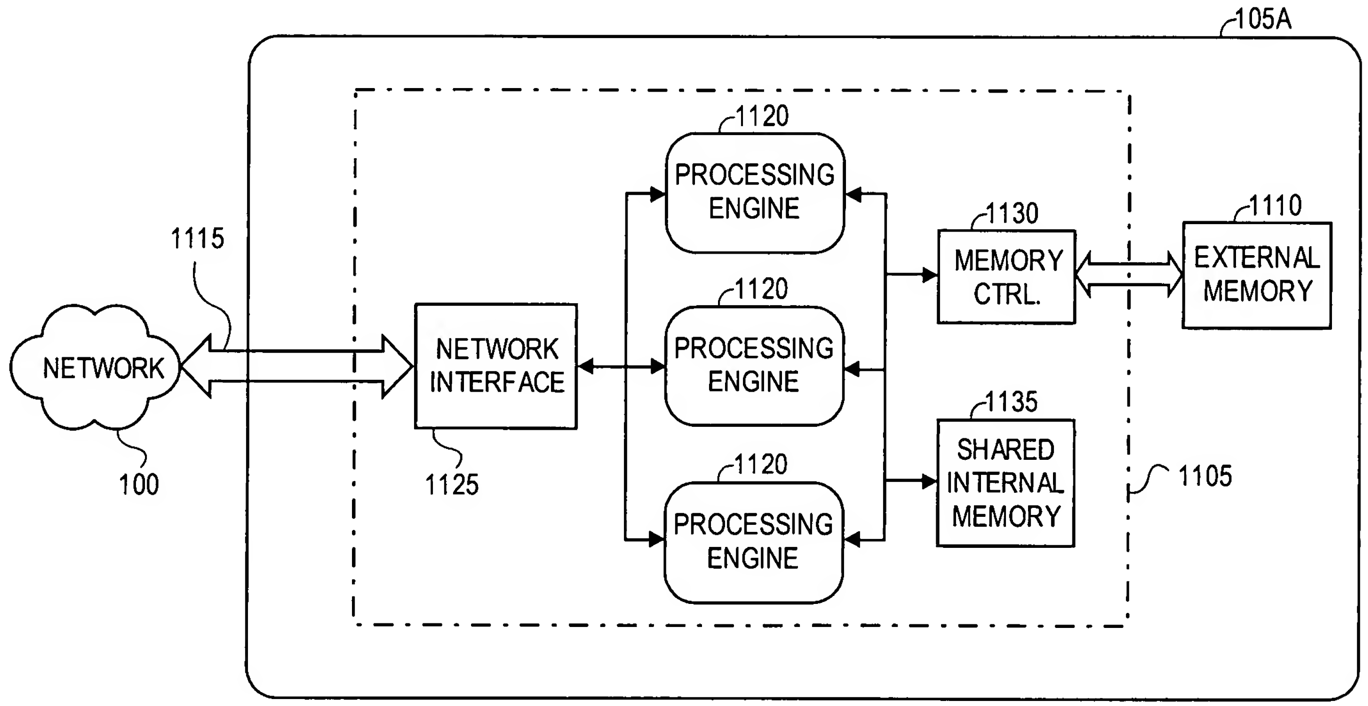
105A
1115
NETWORK
100
NETWORK INTERFACE
1125
1120
PROCESSING ENGINE
1120
PROCESSING ENGINE
1120
PROCESSING ENGINE
1130
MEMORY CTRL.
1135
SHARED INTERNAL MEMORY
1105
1110
EXTERNAL MEMORY

FIG. 11